TITLE:

Parallel Computation and Synchronized Term Rewriting Systems : Extended Abstract (Algebraic Semigroups, Formal Languages and Computation)

AUTHOR(S):

Fujita, Ken-Etsu; Middeldorp, Aart

CITATION:

Fujita, Ken-Etsu ...[et al]. Parallel Computation and Synchronized Term Rewriting Systems : Extended Abstract (Algebraic Semigroups, Formal Languages and Computation). 数理解析研究所講究録 2001, 1222: 105-113

ISSUE DATE:

2001-07

URL:

http://hdl.handle.net/2433/41323

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

数理解析研究所講究録 1222 巻 2001 年 105-113

Parallel Computation and Synchronized Term Rewriting Systems
– Extended Abstract –

Ken-Etsu Fujita (藤田 憲悦)
Shimane University
(島根大学)
fujiken@cis.shimane-u.ac.jp

Aart Middeldorp
University of Tsukuba
(筑波大学)
ami@is.tsukuba.ac.jp

Abstract

We present an extension of term rewriting systems with the mechanism of synchronization, called synchronized TRSs. The notion of synchronization is newly introduced for synchronizing applications of rewrite rules. The construction of synchronized TRSs can be regarded as a combination problem of TRSs. We prove fundamental properties of synchronized ground TRSs:

(1) Termination property is decidable for finite right-ground synchronized TRSs.

(2) The reachability problem for synchronized ground TRSs is undecidable.

We show two proofs of the second result, using the halting problem for two counter automata and using Post's correspondence problem. The proofs also reveal explicitly a role of synchronization to be a correspondence or coordination relation among computations.

1 Introduction

The notion of term rewriting systems (TRSs) gives a mathematical model of computation in terms of a set of rewrite rules. In particular, the traditional TRSs play a role of a natural model of sequential computations. We present an extension of term rewriting systems with the mechanism of synchronization, called synchronized TRSs. Synchronization is widely used as a mechanism of parallel programming with explicit concurrency. The notion of synchronization is newly introduced into TRSs from the motivation for *synchronizing applications of rewrite rules*, i.e., synchronized computation. The new notion captures intrinsically parallel computation. We prove fundamental properties of synchronized TRSs:

(1) Termination is decidable for finite right-ground synchronized TRSs.

(2) The reachability problem for synchronized ground TRSs is undecidable.

It is well-known that termination for finite right-ground TRSs is decidable. We show that this property also holds for finite right-ground synchronized TRSs, although the parallel computation is not closed under contexts. On the other hand, the reachability problem of ground TRSs is known $\mathcal{P}$-complete [TN83], and the decidability result is now extended to more general cases [Oya90, NT99]. However, from the second result the reachability problem becomes undecidable even in the case of synchronized *ground* TRSs. Moreover, synchronized ground TRSs has Turing computability. The encodings used in the proofs also reveal explicitly a role or mechanism of synchronization in pure programming languages.

The notion of synchronization is involved in the work on Metaterm Rewriting Calculus (MRC) by Ohshima and Sakabe [OS99]. Although both have synchronized annotations (labels) for rewrite rules, the key distinction is that occurrences t_1 and t_2 to be synchronously rewritten are explicitly mentioned as a new term $t_1 \mathbf{sync} t_2$ in [OS99], and hence our results obtained here cannot be applied to their calculus straightforwardly.

2 Synchronized Term Rewriting Systems

Since we assume that readers are, more or less, familiar with term rewriting systems (TRSs), first we briefly introduce the concept of a term rewriting system following [Hue80, Klo92, BN98]:

Let $\mathcal{V}$ be a countably infinite set of *variables*. Let $\mathcal{F}$ be a non-empty set of *function symbols* with $\mathcal{V} \cap \mathcal{F} = \emptyset$. Each element in $\mathcal{F}$ is graded by an arity function $\mathrm{a} : \mathcal{F} \to \mathcal{N}$. We define $\mathcal{F}_n = \{f \in \mathcal{F} \mid \mathrm{a}(f) = n\}$. A pair $\Sigma = \langle \mathcal{V}, \mathcal{F} \rangle$ is called a *signature*.

The set $\mathcal{T}$ of *terms* of a first-order language over Σ is defined as the free $\mathcal{F}$-algebra generated by $\mathcal{V}$. That is, a term is either a variable or in the form of $f(t_1, \ldots, t_n)$ for some $f \in \mathcal{F}_n$ and $t_1, \ldots, t_n \in \mathcal{T}$. Terms containing no variables are called *ground terms*.

A *rewrite rule* is a pair $\langle s, t \rangle$ of terms imposed by the following conditions:

(1) the term s in the left-hand side is not a variable, and

(2) the variables appeared in the right-hand side t are contained in the left-hand side s.

We write $s \to t$ for $\langle s, t \rangle$. Let R be a set of rewrite rules. *A term rewriting system* is defined as a pair $\langle \Sigma, R \rangle$. For simplicity we often call a term rewriting system any set R of rewrite rules. A term rewriting system R is called *finite* if R is a finite set. If for each $l \to r \in R$ the right-hand side r contains no variables, then the term rewriting system R is called *right-ground*.

A *term context* is obtained from a term by replacing one occurrence of the subterm with a special symbol [ ] called a hole. A term context is denoted by $t[\,]$. For a term context $t[\,]$ and a term s, $t[s]$ denotes a term obtained by placing s into the hole of $t[\,]$. A *substitution* σ is a map from $\mathcal{T}$ to $\mathcal{T}$ such that $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$ for every $f \in \mathcal{F}_n$.

A *one-step rewriting (reduction) relation* $\to_R$ is defined as $t[\sigma(l)] \to_R t[\sigma(r)]$ for some $l \to r \in R$. We say that the term $t[\sigma(l)]$ is *reduced (contracted)* to $t[\sigma(r)]$ by the application of $l \to r \in R$, and $\sigma(l)$ is called a *redex*. A term is in *normal form* if and only if it contains no redex. We denote the transitive closure of $\to_R$ by $\to_R^+$, and the reflexive and transitive closure by $\to_R^*$, respectively.

Second we define a *synchronized term rewriting system (STRS)* to be a term rewriting system consisting of labelled rewrite rules. We write $i : l \to r$ for a labelled rewrite rule, where i is called a label or an annotation for synchronization. An STRS R is called *finite right-ground* if R is a finite set and the right-hand side of every rule in R contains no variables. *A ground STRS* is an STRS in which no rewrite rule contains variables. *A one-step rewriting relation* $\rightrightarrows_R$ *by an STRS* R is defined as contracting, at pairwise disjoint positions in a term, maximal number of redexes simultaneously by rewrite rules only with the same annotation. The transitive closure of $\rightrightarrows_R$ is denoted by $\rightrightarrows_R^+$, and the reflexive and transitive closure is by $\rightrightarrows_R^*$. We call computation by the application of $\rightrightarrows_R$ *maximally parallel computation.*

For instance, let R be the following STRS:

$$R = \begin{pmatrix} 1 : a \to b & 2 : b \to c \\ 1 : a \to c & 2 : c \to a \\ 1 : f(x) \to g(x, x) & \end{pmatrix}$$

Then we have one step reduction by maximally parallel computation, as follows:
$f(a) \rightrightarrows_R f(b)$, $f(c)$, or $g(a, a)$;
$g(a, f(a)) \rightrightarrows_R g(b, f(b))$, $g(b, f(c))$, $g(c, f(b))$, $g(c, f(c))$, $g(b, g(a, a))$, or $g(c, g(a, a))$;
$g(a, b) \rightrightarrows_R g(b, b)$, $g(c, b)$, or $g(a, c)$.

¿From the definition, the maximally parallel computation is not closed under a context in general. That is, for a term context $t[\ ]$, $t[s_1] \not\rightrightarrows_R t[s_2]$ even though $s_1 \rightrightarrows_R s_2$. In this case, we only obtain $t[s_1] \rightrightarrows_R t'[s_2]$ for some context $t'[\ ]$. It is clear from the definition that $\rightrightarrows_R \subseteq \to_{|R|}^+$, where $|R|$ denotes a TRS obtained by omitting the labels from R. The inverse direction $\rightrightarrows_R^+ \supseteq \to_{|R|}^+$ does not hold true, in general, because of the synchronized annotations. This property is an essential point of giving encodings in the following sections, so that the correspondence relation of Post's correspondence problem is naturally represented as the notion of synchronization together with ground terms as well as the coordination of the two counters of automata. In the following sections, we investigate the termination and reachability properties for STRSs with the maximally parallel computation.

3 Termination for finite right-ground STRSs

In this section, we prove that termination is decidable for finite right-ground STRSs. An STRS R is *terminating (strongly normalizing or noetherian)* if and only if there is no infinite reduction sequence $t_1 \rightrightarrows_R t_2 \rightrightarrows_R \cdots$. An STRS is *weakly normalizing* if every term has a normal form.

Lemma 1 *An* STRS *is not terminating if and only if it admits infinite reduction sequences that contain a root rewrite step.*

Proof. We show the if part since the only-if part is clear. Assume that we have an infinite reduction sequence *Seq*: $t_1 \rightrightarrows t_2 \rightrightarrows \cdots$. By induction on the structure of t_1, we prove that there also exists an infinite reduction sequence that contain a root rewrite step.

The base case is obvious, i.e., a constant t_1 itself is contracted.

Suppose $t_1 = f(s_1, \ldots, s_n)$. If *Seq* contains no root rewrite steps, then all the steps take place in the proper subterms $s_1, \ldots, s_n$. The number of the subterms is finite, while *Seq* admits an infinite number of rewrite steps. Hence, from pigeonhole principle there exists s_i that admits an infinite reduction. According to the induction hypothesis, we also have an infinite reduction that contains a root rewrite step. □

Corollary 1 *A right-ground* STRS R *is terminating if and only if for any rule* $i : l \to r \in R$, r *is terminating.*

Proof. We show the only-if part since the if part is clear. Suppose that R had an infinite reduction sequence. From Lemma 1, the infinite path contains a root rewrite step. A right-ground STRS guarantees that a redex is the right-hand side of a rewrite rule. Now all of them are terminating, which gives a contradiction. □

Lemma 2 *Let* R *be a finite right-ground* STRS.
R *is not terminating if and only if for some term context* $t[\,]$ *and a rewrite rule* $i : l \to r \in R$, *we have* $r \rightrightarrows_R^+ t[r]$.

Proof. The if part is proved by induction on the cardinality of R. If R is not terminating, then from Corollary 1 there exists $i : l \to r \in R$ such that we have an infinite sequence *Seq*: $r \equiv t_1 \rightrightarrows_R t_2 \rightrightarrows_R \cdots$. According to the excluded middle, the rewrite rule $i : l \to r \in R$ is applied in *Seq* or not.
Case where $i : l \to r$ is applied in *Seq*:
We have $r \rightrightarrows_R^* t'[\sigma(l)] \rightrightarrows_R t[r]$ for some term contexts $t'[\,]$ and $t[\,]$.
Otherwise:
The statement can be confirmed by the use of the induction hypothesis for $R \backslash \{i : l \to r\}$.

For the only-if part, if we have $r \rightrightarrows_R^+ t[r]$, then we also have $t[r] \rightrightarrows_R^+ t_1[t[r]] \rightrightarrows_R^+ \cdots$ for some context $t_1[\,]$, even though the computation is not closed under contexts. Hence, r induces an infinite reduction sequence, and R is not terminating. □

Proposition 1 *Termination is decidable for finite right-ground* STRS*s.*

Proof. The decision procedure for termination of finite right-ground STRSs is derived from Lemma 2. Generate the reducts of the right-hand sides of all the rewrite rules in a breadth-first way. If we do not observe $r \rightrightarrows_R^+ t[r]$ for some context $t[\,]$, then the STRS is terminating from Lemma 2, and hence the process of the reductions eventually terminates. If we observe $r \rightrightarrows_R^+ t[r]$ for some context $t[\,]$, then the STRS cannot be terminating from Lemma 2. □

We are currently investigating whether other known (un)decidability results for restricted classes of TRSs carry over to STRSs.

4 Synchronization and Post's Correspondence Problem

In this section and the following section, we prove that the reachability property is undecidable for STRSs even consisting only of ground terms, and that ground STRSs has Turing computability. The decidable reachability property cannot be carried over to ground STRSs from ground TRSs. The *reachability problem* for STRSs is, given an STRS R and terms t_1 and t_2, to decide whether $t_1 \rightrightarrows_R^* t_2$ or not.

Let Σ be a non-empty finite set including at least two symbols, called alphabet. For simplicity, we set Σ as $\{0,1\}$ without loss of generality. Let Σ^+ be a set of all words over Σ excluding null words. Given an arbitrary finite set $\{(\alpha_i, \beta_i) \mid \alpha_i, \beta_i \in \Sigma^+ \ (1 \leq i \leq n)\}$ of pairs of corresponding non-null words on Σ, then *Post's correspondence decision problem (PCP)* [Pos46] is to decide if there exist $i_1, i_2, \ldots, i_k$ such that $\alpha_{i_1}\alpha_{i_2}\cdots\alpha_{i_k} = \beta_{i_1}\beta_{i_2}\cdots\beta_{i_k}$ $(k \geq 1, 1 \leq i_j \leq n)$.

Given an instance of Post's correspondence problem, then define the following STRS R over the signature consisting of constants a, b, c, d, unary function symbols $0, 1$, and a binary function symbol f:

$$R = \begin{pmatrix} 1 : a \to \alpha_1(a) & \cdots\cdots & n : a \to \alpha_n(a) \\ 1 : b \to \beta_1(b) & \cdots\cdots & n : b \to \beta_n(b) \\ n+1 : 0(a) \to c & & n+2 : 1(a) \to c \\ n+1 : 0(b) \to d & & n+2 : 1(b) \to d \\ n+3 : 0(c) \to c & & n+4 : 1(c) \to c \\ n+3 : 0(d) \to d & & n+4 : 1(d) \to d \end{pmatrix}$$

where the natural numbers $1, 2, \ldots, n+4$ are used as annotations for synchronization.

Lemma 3 *$f(a,b) \rightrightarrows_R^* f(c,d)$ if and only if the instance of Post's correspondence problem has a solution.*

Proof. ($\Rightarrow$) The term $f(a,b)$ can be reduced only by the use of rewrite rules annotated with $1 \leq j \leq n$. Then we obtain

$$f(a,b) \rightrightarrows_R^+ f(\alpha_{i_1}(\alpha_{i_2}\cdots\alpha_{i_k}(a)\cdots), \beta_{i_1}(\beta_{i_2}\cdots\beta_{i_k}(b)\cdots))$$

for some $i_1, i_2, \ldots, i_k$. Now one can apply only rewrite rules annotated with either $n+1$ or $n+2$, and then apply only the rules annotated with either $n+3$ or $n+4$. The right-hand side is here reduced to $f(c,d)$ from the assumption, which implies that $\alpha_{i_1}\alpha_{i_2}\cdots\alpha_{i_k} = \beta_{i_1}\beta_{i_2}\cdots\beta_{i_k}$ from the definition of rules annotated with $n+1, n+2, n+3, n+4$.

($\Leftarrow$) Assume that $\alpha_{i_1}\alpha_{i_2}\cdots\alpha_{i_k} = \beta_{i_1}\beta_{i_2}\cdots\beta_{i_k}$ for some $i_1, i_2, \ldots, i_k$. We now have

$$f(a,b) \rightrightarrows_R^+ f(\alpha_{i_1}(\alpha_{i_2}\cdots\alpha_{i_k}(a)\cdots), \beta_{i_1}(\beta_{i_2}\cdots\beta_{i_k}(b)\cdots)).$$

Then from the synchronized applications of rewrite rules annotated with either $n+1$ or $n+2$, following with either $n+3$ or $n+4$, one eventually has

$$f(\alpha_{i_1}(\alpha_{i_2}\cdots\alpha_{i_k}(a)\cdots), \beta_{i_1}(\beta_{i_2}\cdots\beta_{i_k}(b)\cdots)) \rightrightarrows_R^+ f(c,d),$$

since $\alpha_{i_1}\alpha_{i_2}\cdots\alpha_{i_k} = \beta_{i_1}\beta_{i_2}\cdots\beta_{i_k}$. □

Theorem 1 *The reachability problem for ground STRSs is undecidable.*

Proof. From Lemma 3 and [Pos46]. □

The theorem above says that it is unsolvable to determine whether the term $f(a,b)$ has a normal form. In other words, it is undecidable to decide whether the STRS R constructed from PCP is weakly normalizing or not.

5 Synchronization and Two-Counter Automata

5.1 Two Counter Automata

Following [HU79, Sch97], we briefly introduce two counter automata. O denotes an operation on counters, and T stands for a states of counters, as follows:

$O \in \mathcal{O} = \{NOP, INC, DEC\}$: operations on counters
$T \in \mathcal{T} = \{Z, NZ\}$: states of a counter

A *two-counter automaton* is defined by an off-line Turing machine with two counters, as follows:

- Two Counter Automaton $\langle \mathcal{Q}, q_s, q_f, \delta, C_1, C_2 \rangle$
  $\mathcal{Q}$: a set of states including an initial state q_s and a final state q_f
  δ: a next-step deterministic function from $\mathcal{T}^2 \times \mathcal{Q}$ to $\mathcal{O}^2 \times \mathcal{Q}$ such that
  $\langle T_1, T_2, q \rangle \mapsto_\delta \langle O_1, O_2, q' \rangle$
  C_1, C_2: two counters denoting natural numbers

An instantaneous description of the two counter automaton is defined as $\langle C_1, C_2, q \rangle$, as follows:

- Instantaneous description of the automaton (ID)
  $\langle n_1, n_2, q \rangle \mapsto_\delta \langle n_1', n_2', q' \rangle$ where $n_i, n_i' \in \mathcal{N}$
  Starting ID: $\langle 0, 0, q_s \rangle$; Final ID: $\langle 0, 0, q_f \rangle$

For a technical reason, without loss of generality we have the following assumption (A):
If $\langle T_1, T_2, q_1 \rangle \mapsto_\delta \langle O_1, O_2, q_2 \rangle$, then there is no transition such that
$\langle T_1', T_2', q_1 \rangle \mapsto_\delta \langle O_1', O_2', q_2 \rangle$ for $T_1 \neq T_1'$, $T_2 \neq T_2'$, $O_1 \neq O_1'$, or $O_2 \neq O_2'$.
Remarked that the assumption guarantees that $\langle T_1, T_2, q \rangle \mapsto_\delta \langle O_1, O_2, q' \rangle$ is uniquely determined by the pair of the two states (q, q').

Theorem 2 ([Min61, Fis66, HU79])
A two-counter automaton can simulate an arbitrary Turing machine.

5.2 Encoding of C_2-Automata via Synchronized Ground TRS

It is, in general, impossible to encode two-counter automata via a ground TRS, because one has Theorems 2 while ground TRSs have less computational power than Turing machine.

First we observe what is the essential distinction between two-counter automata and ground TRSs. Filling the deep gap naturally leads to the notion of synchronized TRSs.

Following [Sch97], we use one binary-function symbol f, and constants $0, 1$ and q_j^1, q_j^2 for $q_j \in \mathcal{Q}$, where $1 \leq j \leq \mathtt{card}(\mathcal{Q})$. For a set $\mathcal{Q}$, $\mathtt{card}(\mathcal{Q})$ denotes the cardinality of the set.

- Code of the ID $\langle n, n', q\rangle$ denoted by $t_{\langle n,n',q\rangle}$:

$$t_{\langle n,n',q\rangle} = f(\, f(0, \underbrace{f(1, \cdots f(1}_{n \text{ times}}, q^1) \cdot \cdot)),\ f(0, \underbrace{f(1, \cdots f(1}_{n' \text{ times}}, q^2) \cdot \cdot))\,)$$

- Code of the transition $\delta : \langle T_1, T_2, q_1\rangle \mapsto \langle O_1, O_2, q_2\rangle$

Let $k = \mathtt{card}(\delta)$. For $j \in \{1, \ldots, k\}$, δ is coded as a union of the two sets of the rewrite rules

$$\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 = \{b_j^1 \to b_j'^1 \mid j = 1, \ldots, k\} \cup \{b_j^2 \to b_j'^2 \mid j = 1, \ldots, k\}$$

consisting of $b_j^i \to b_j'^i$ $(i = 1, 2)$ which are defined based on (A), as follows:

$T_i \backslash O_i$	DEC	NOP	INC
Z		$f(0, q_1^i) \to f(0, q_2^i)$	$f(0, q_1^i) \to f(0, f(1, q_2^i))$
NZ	$f(1, q_1^i) \to q_2^i$	$f(1, q_1^i) \to f(1, q_2^i)$	$f(1, q_1^i) \to f(1, f(1, q_2^i))$

Here, $\mathcal{R}_i$ describes a computation relating to the counter i $(i = 1, 2)$. Now we can simulate each transition of ID's

$$\langle 0, 0, q_s\rangle \mapsto_\delta \langle n, n', q\rangle \mapsto_\delta \cdots$$

via the two steps rewriting denoted by $\to_{\mathcal{R}}^2$

$$t_{\langle 0,0,q_s\rangle} \to_{\mathcal{R}}^2 t_{\langle n,n',q\rangle} \to_{\mathcal{R}}^2 \cdots.$$

Lemma 4 *If we have* $\langle n_1, n_2, q_1\rangle \mapsto_\delta \langle n_1', n_2', q_2\rangle$, *then* $t_{\langle n_1,n_2,q_1\rangle} \to_{\mathcal{R}}^2 t_{\langle n_1',n_2',q_2\rangle}$.

Proof. From the case analysis on the definition of the implementation of δ. □

Proposition 2

A two-counter automaton stops, then $f(f(0, q_s^1), f(0, q_s^2)) \to_{\mathcal{R}}^* f(f(0, q_f^1), f(0, q_f^2))$.

Proof. From Lemma 4. □

Noted that the inverse direction of Proposition 2 is not true in general. For instance, take the following δ:

$$\delta = \left\{ \begin{array}{ll} \langle Z, Z, q_s\rangle \mapsto_{\delta_1} \langle INC, NOP, q_1\rangle, & \\ \langle NZ, NZ, q_1\rangle \mapsto_{\delta_2} \langle NOP, NOP, q_2\rangle, & \langle NZ, Z, q_1\rangle \mapsto_{\delta_3} \langle NOP, INC, q_3\rangle, \\ \langle NZ, NZ, q_2\rangle \mapsto_{\delta_4} \langle DEC, DEC, q_f\rangle, & \langle NZ, NZ, q_3\rangle \mapsto_{\delta_5} \langle DEC, DEC, q_f\rangle \end{array} \right\}$$

Then we obtain the expected transitions:

$$\langle 0, 0, q_s\rangle \mapsto_{\delta_1} \langle 1, 0, q_1\rangle \mapsto_{\delta_3} \langle 1, 1, q_3\rangle \mapsto_{\delta_5} \langle 0, 0, q_f\rangle$$

However, not only this but also the following can be obtained by the rewriting $\to_{\mathcal{R}}$:

$$\begin{array}{l} f(f(0, q_s^1), f(0, q_s^2)) \to_{\mathcal{R}}^2 f(f(0, f(1, q_1^1)), f(0, q_1^2)) \\ \to_{\mathcal{R}} f(f(0, f(1, q_2^1)), f(0, q_1^2)) \to_{\mathcal{R}} f(f(0, f(1, q_2^1)), f(0, f(1, q_3^2))) \\ \to_{\mathcal{R}} f(f(0, q_f^1), f(0, f(1, q_3^2))) \to_{\mathcal{R}} f(f(0, q_f^1), f(0, q_f^2)) \end{array}$$

which cannot give an ID. In order to obtain the inverse direction, it is not enough to make a simple union of the two rewriting systems; we need a mechanism of *synchronization between the two rewrite rules*, $b_j^1 \to b_j'^1$ and $b_j^2 \to b_j'^2$. Now we define a synchronized ground TRS $\mathcal{R}^s$ over $\mathcal{R}_1$ and $\mathcal{R}_2$, as follows:

$$\mathcal{R}^s = \{s_j : b_j^1 \to b_j'^1 \mid j = 1, \ldots, k\} \cup \{s_j : b_j^2 \to b_j'^2 \mid j = 1, \ldots, k\},$$

where the rewrite rules with the same annotation s_j can be *applied in parallel*, which means that the synchronization of the two counters can be implemented by synchronous rewriting over $\mathcal{R}_1$ and $\mathcal{R}_2$.

It is remarked that in Schubert's encoding via second-order unification [Sch97, Sch98], the coordination equation $\langle s_4, u_4 \rangle$ plays a role of the synchronization for the two counters.

We now have the inverse direction of Proposition 2 in the following sense:

Proposition 3
If $f(f(0, q_s^1), f(0, q_s^2)) \rightrightarrows_{\mathcal{R}^s}^* f(f(0, q_f^1), f(0, q_f^2))$, *then a two-counter automaton stops.*

Proof. If we have $t_{\langle n_1, n_2, q_1 \rangle} \rightrightarrows_{\mathcal{R}^s} t_{\langle n_1', n_2', q_2 \rangle}$, then $\langle n_1, n_2, q_1 \rangle \mapsto_\delta \langle n_1', n_2', q_2 \rangle$. □

Theorem 3 *A ground STRS can simulate an arbitrary Turing machine.*
The reachability problem is undecidable for ground STRSs.

Proof. We can also establish the same statement as Proposition 2 with respect to $\rightrightarrows_{\mathcal{R}^s}^*$, and from Proposition 3 and Theorem 2. □

References

[BN98] Baader, R. and T. Nipkow: Term *Rewriting* and *All That*, Cambridge University Press, 1998.

[Fis66] Fischer, P. C.: Turing Machines with Restricted Memory Access, *Information and Control* 9, pp. 364–379, 1966.

[Hue80] Huet, G. : Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems, *Journal of the Association for Computing Machinery*, Vol. 27, No. 4, pp. 797–821, 1980.

[HU79] Hopcroft, J. E. and J. D. Ullman: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.

[Klo92] Klop, J. W.: Term Rewriting Systems, *Handbook of Logic in Computer Science* Vol. 2, Oxford University Press, pp. 1–116, 1992.

[Min61] Minsky, M. L.: Recursive Unsolvability of Post's Problem of "TAG" and Other Topics in Theory of Turing Machines, *Annals of Mathematics*, Vol. 74, No. 3, pp. 437–455, 1961.

[NT99] Nagaya, T. and Y. Toyama: Decidability for left-linear growing term rewriting systems, Lecture Notes in Computer Science 1631, pp. 256-270, 1999.

[Oya90] Oyamaguchi, M.: The Reachability and Joinability Problems for Right-Ground Term-Rewriting Systems, *Journal of Information Processing*, Vol. 13, No. 3, pp. 347–354, 1990.

[OS99] Ohshima, A. and T. Sakabe: Design of the Interpreter for Metaterm Rewriting Calculus with Dynamic Discrimination Nets, *Technical Report of IEICE*, SS99–61, pp. 9–16, 1999 (*in Japanese*).

[Pos46] Post, E. L.: A variant of a recursively unsolvable problem, *Bulletin of the American Mathematical Society*, Vol. 52, No. 4, pp. 246–268, 1946.

[Sch97] Schubert, A.: Second-order unification and type inference for Church-style, Tech. Report TR 97-02 (239), Institute of Informatics, Warsaw University, January 1997.

[Sch98] Schubert, A.: Second-order unification and type inference for Church-style, *Proc. ACM Symposium on Principles of Programming Languages*, pp. 279–288, 1998.

[TN83] Togashi, A. and S. Noguchi: Some Decision Problems and Their Time Complexity for Term Rewriting Systems, *The Institute of Electronics, Information and Communication Engineers Transactions on Information and Systems*, Vol. J66–D, No. 10, pp. 1177–1184, 1983 (*in Japanese*).